Chương 2.

1. Ràng buộc toàn vẹn là gì?

Tập các quy tắc mà mọi dữ liệu trong CSDL phải tuân theo nhằm đảm bảo tính toàn vẹn ( để đảm bảo dữ liệu luôn luôn đúng) của cơ sở dữ liệu.

1. Tại sao các quan hệ phải có ràng buộc toàn vẹn?

nhằm đảm bảo tính toàn vẹn ( để đảm bảo dữ liệu luôn luôn đúng) của cơ sở dữ liệu

1. Khóa chính (primary key) của một quan hệ Q

**Khóa chính (primary key):** Một trong các khóa ứng viên sẽ được chọn làm **khóa chính** cho quan hệ.

1. Khóa ngoại (foreign key) của một quan hệ Q

Một hay nhiều thuộc tính (khóa chính hay thuộc tính có ràng buộc unique) trong một quan hệ A xuất hiện trong một quan hệ B khác được gọi là khóa ngoại trong quan hệ B.

1. Ràng buộc về miền giá trị của 1 thuộc tính

các giá trị trong cột phải nằm trong miền giá trị của cột.

1. Một thuộc tính có ràng buộc Unique?

mỗi giá trị trong cột có ràng buộc này phải là duy nhất trong cột đó.

1. Một thuộc tính có ràng buộc not null nghĩa là?

Các ô trong cột có ràng buộc này bắt buộc phải có giá trị khác null

1. Một thuộc tính có ràng buộc Check (điều kiện) nghĩa là?

Là thuộc tính phải thỏa điều kiện check của rảng buộc đó

1. Câu lệnh khai báo, thêm và xóa các ràng buộc của quan hệ Q.

Khai báo ở mức cột :

+ col datatype Check(condition)

Khai báo ở mức bảng:

+CONSTRAINT name\_constrain check( condition)

Thêm:

ALTER TABLE table\_name ADD CONSTRAINS condition\_name condition(col) ( tuỳ chọn áp dụng cho mức cột hay bảng )

+ Xoá:

ALTER TABLE table\_name DROP CONSTRAINS condition\_name

1. Cho Q1(A int, B int), Q2(A int, C int). A trong Q2 là khóa ngoại tham chiếu qua A của Q1. Thao tác nào dưới đây không thực hiện được
2. Thêm 1 dòng có A = 1 vào Q2, nhưng trong Q1 chưa có dòng nào có A = 1.
3. Sửa 1 dòng trong Q2 có A = 1 thành A=0, nhưng trong Q1 chưa có dòng nào có A = 0.
4. Cả (1) và (2) đều đúng
5. Cả (1) và (2) đều sai
6. Trigger là gì?

Trigger là một loại stored procedure đặc biệt (không có tham số) được thực thi (execute) một cách tự động khi có một sự kiện thay đổi dữ liệu (data modification)

1. Ý nghĩa của biến cố, điều kiện và hành động trong trigger

- Trigger chỉ được gọi khi xảy ra một số sự kiện nhất định do người lập trình cơ sở dữ liệu chỉ định (thường là chèn, xoá, cập nhật)

- Sau khi được gọi bởi sự kiện kích hoạt của nó, trình kích hoạt sẽ kiểm tra một điều kiện

- Nếu điều kiện của trình kích hoạt được thoả mãn, thì hành động được liên kết với trình kích hoạt sẽ được thực hiện bởi DBMS

1. Ý nghĩa của For và After trong câu lệnh tạo trigger

FOR: trigger sẽ được thực hiện **trước** khi thay đổi dữ liệu thực sự được áp dụng vào cơ sở dữ liệu. "FOR" mà không có "AFTER" hoặc "INSTEAD OF" đi kèm, thì **"AFTER"** được coi là giá trị mặc định

AFTER:

+chỉ định rằng DML trigger chỉ được kích hoạt khi tất cả các hoạt được chỉ định trong phần sql\_statement trigger đã thực thi thành công. Tất cả các hành động **referential cascade**  và **constraint checks** cũng phải thành công trước khi trigger kích hoạt.

+ không thể sử dụng AFTER cho mức **view**

1. Ý nghĩa của câu lệnh dưới

Create trigger AA on Q after delete as declare @new int, @old int select @old=ol.B from deleted ol if (@old >2) print ('Notify Customer Relations');

Tạo ra trigger tên AA trong table Q, sau khi có hành động delete trong Q, trigger được thực hiện --> tạo ra hai biến old và new, old = những cái vừa bị xoá trong cột B,nếu old > 2 thì xuất ra thông báo

1. Instead of trigger là gì?

-instead of: Chỉ định rằng trình kích hoạt DML được thực thi thay vì câu lệnh SQL kích hoạt, do đó, ghi đè hành động của câu lệnh kích hoạt. INSTEAD OF không thể được chỉ định cho DDL hoặc trình kích hoạt đăng nhập. Tối đa, một trình kích hoạt INSTEAD OF cho mỗi câu lệnh INSERT, UPDATE hoặc DELETE có thể được xác định trên một bảng hoặc dạng xem. Trình kích hoạt INSTEAD OF không được phép trên các chế độ xem có thể cập nhật sử dụng WITH CHECK OPTION.

- DML trigger thường được sử dụng để thực thi các quy tắc và dữ liệu kinh doanh

1. Ý nghĩa của câu lệnh dưới

Create trigger AA on Q after delete as declare @old int select @old=ol.B from deleted ol if (@old >2) Begin print ('Notify Customer Relations'); Rollback; end

Tạo ra trigger tên AA trong bảng Q, trigger thực hiện sau khi thực hiện lệnh xoá trong bảng Q , khai báo biến old bằng cái vừa bị xoá trong cột B, nếu old > 2 thì in ra thông báo 'Notify Customer Relations' và khôi phục dữ liệu vừa xoá.

1. Cho

Create trigger AA on Q after delete as declare @old int select @old=ol.B from deleted ol if (@old >2) Begin print ('Notify Customer Relations'); Rollback; end

Từ khóa “after” trong câu lệnh trên có nghĩa là gì?

Khi biến cố delete xảy ra thì trigger sql statement là declare @old int select @old=ol.B from deleted ol if (@old >2) Begin print ('Notify Customer Relations'); Rollback; end , khi thực hiện thành công và không bị rollback thì delete ( dml trigger ) mới được chính thức thực hiện.

1. Cho

Create trigger AA on Q for delete as declare @old int select @old=ol.B from deleted ol if (@old >2) Begin print ('Notify Customer Relations');

Từ khóa “for” trong câu lệnh trên có nghĩa là gì?

Từ khoá for có nghĩa khi lệnh delete được thực hiện. Trigger cũng đồng thời được kích hoạt Sql statement ( khai báo bién old bằng cái vừa bị xoá trong cột B nếu > 2 thì in ra 'Notify Customer Relations , cuối cùng dữ liệu bị xoá từ lệnh delete chính thức tác động lên bảng Q.

1. Ý nghĩa của câu lệnh dưới

Create trigger BB on Q1 instead of insert as Declare @newA int, @newB int Select @newA = n.A, @newB = n.B From Inserted n Insert into Q2 values(@newA, @newB)

+ khi có một lệnh insert và bảng Q1 được thực hiện trên bảng Q1 thì trigger được kích hoạt thay vì lệnh insert vào bảng Q1.

+ Khai báo hai biến newA và newB kiểu dữ liệu int

+ Thực hiện truy vấn để newA , newB lần lượt bằng giá trị cột "A", "B" trong bảng Inserted ( bảng chứa dữ liệu tạm thời sẽ được insert vào Q1)

+ Thực hiện lệnh Insert để chèn hai giá trị newA,newB vào bảng Q2.

1. View là gì?

Dựa vào thông tin trên, một view là một bảng ảo trong cơ sở dữ liệu. Nó không lưu trữ dữ liệu thực tế mà thay vào đó lấy dữ liệu từ một hoặc nhiều bảng gốc (base tables) hoặc các view khác trong cơ sở dữ liệu thông qua một truy vấn. View cho phép bạn xem và truy vấn dữ liệu từ các bảng gốc một cách thuận tiện bằng cách tạo một giao diện ảo với cấu trúc cột và dòng được xác định bởi truy vấn đó.

1. Mục đích của sử dụng view

* **Tập trung, Đơn giản hóa và Tùy chỉnh Quan điểm Người dùng:**

Chế độ xem cho phép bạn tùy chỉnh và đơn giản hóa cách mỗi người dùng tương tác với cơ sở dữ liệu. Những người dùng hoặc ứng dụng khác nhau có thể có các chế độ xem phù hợp riêng, chỉ hiển thị dữ liệu liên quan, giúp làm việc với cơ sở dữ liệu dễ dàng hơn.

* **Cơ chế Bảo mật**: Chế độ xem có thể đóng vai trò là cơ chế bảo mật bằng cách cấp cho người dùng quyền truy cập vào dữ liệu thông qua chế độ xem trong khi hạn chế hoặc không cấp cho họ quyền trực tiếp để truy cập vào các bảng cơ sở bên dưới. Điều này cho phép kiểm soát chi tiết hơn việc truy cập dữ liệu và giúp bảo vệ thông tin nhạy cảm.
* **Giao diện Tương thích ngược**: Các khung nhìn có thể được sử dụng để cung cấp giao diện tương thích ngược khi lược đồ của bảng đã thay đổi. Thay vì sửa đổi các truy vấn và mã hiện có, bạn có thể cập nhật dạng xem để trình bày dữ liệu theo cách duy trì khả năng tương thích với các ứng dụng hoặc truy vấn hiện có.

1. Trong SQL server, cú pháp câu lệnh tạo, xóa, sửa view?

Declare by:

CREATE VIEW view\_name AS  
 SELECT column\_name(s)  
 FROM table\_name  
 WHERE condition;

Drop by:

DROP VIEW <name>

Modify by:

ALTER VIEW view\_name AS  
 SELECT column\_name(s)  
 FROM table\_name  
 WHERE condition;

1. Phân loại View?

View và Un-updatable Views

1. Ý nghĩa của câu lệnh

Create view cc as Select A, max(B) as ln from Q1 group by A;

Truy vấn nhóm các bộ theo thuộc tính A, và tìm max thuộc tính B ở mỗi nhóm, tạo một view cc có hai cột gồm A và B các thuộc tính A và max thuộc tính B ở mỗi nhóm trong Q1.

1. Cho PBan(MaPB int, TenPB char(20), MaNQL int). Viết câu lệnh thêm ràng buộc đảm bảo mỗi phòng ban có 1 người quản lý

ALTER TABLE Pban ADD CONSTRAINT CHECK (MaNQL IS NOT NULL).

1. Cho NV(MaNV int, Hoten char(30), Luong int) và câu lệnh alter table Emp add constraint chk\_luong\_min check (salary > 10000). Giải thích ý nghĩa của câu lệnh này.

Thêm vào bảng NV một constraints tên chk\_luong\_min và thực hiện ràng buộc salary trong bảng NV phải lớn hơn 10000.

1. Cho NV(MaNV int, Hoten char(30), Luong int, MaPB int) và view được định nghĩa như sau Create view V\_NV(MaPB, SoNV, TongLg, LgTB) as Select MaPB, Count(\*), Sum(Luong), Avg(Luong) From NV Group by MaPB;

Các câu SQL nào thực hiện được trên view V\_NV

1. Select \* From V\_NV
2. Delete from V\_NV Where MaPB = 1
3. Cả (1) và (2) đều đúng
4. Cả (1) và (2) đều sai

Assignment

**Exercise 1**. What is a foreign key constraint? Why are such constraints important? What is referential integrity?  
*Ràng buộc khóa ngoài* (**a foreign key constrain**)(còn được gọi là *ràng buộc tham chiếu* hoặc *ràng buộc toàn vẹn tham chiếu* ) : Các ràng buộc FOREIGN KEY thực thi tính toàn vẹn tham chiếu ( referential integrity ), về cơ bản nói rằng nếu giá trị cột A tham chiếu đến giá trị cột B thì giá trị cột B phải tồn tại hoặc A là null.

Các ràng buộc (constraints) quan trọng vì chúng cho phép người thiết kế chỉ định ngữ nghĩa của dữ liệu trong cơ sở dữ liệu. *Ràng buộc* là các quy tắc buộc DBMS phải kiểm tra xem dữ liệu có thỏa mãn ngữ nghĩa hay không.

*Tính toàn vẹn tham chiếu* (**referential integrity**)l *yêu* cầu khóa ngoại phải có khóa chính phù hợp hoặc nó phải là null. Ràng buộc này được chỉ định giữa hai bảng (cha và con); nó duy trì sự tương ứng giữa các hàng trong các bảng này. Nó có nghĩa là tham chiếu từ một hàng trong bảng này sang bảng khác phải hợp lệ.

**Exercise 2.** Answer each of the following questions briefly. The questions are based on the following relational schema:

Emp(*eid:* integer, *ename:* string, *age:* integer, *salary:* real)  
Works(*eid:* integer, *did:* integer, *pct\_time:* integer)  
Dept(*did:* integer, *dname*: string, *budget:* real, *managerid:* integer)

1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

- Một số ví dụ về ràng buộc khoá ngoại liên quan đến mối quan hệ Dept:

* Các giá trị trong cột did phải có ràng buộc not null và unique
* Các giá trị trong cột did của relation Work phải có trong cột did của relation Dept
* Managerid trong relation Dept phải có trong cột eid của relation Emp hoặc là bằng null.

- Các tùy chọn để thực thi ràng buộc này là gì khi người dùng cố gắng xóa một bộ dữ liệu Phòng?

* The Default Policy: từ chối xoá một bộ dữ liệu Dept mà có did được tham chiếu bởi khoá ngoại did của relation Works.
* The Cascade Policy: Theo chính sách này, các thay đổi đối với (các) thuộc tính did ( thuộc tính được tham chiếu ) sẽ được mô phỏng theo khoá ngoại ( khoá ngoại did của relation Work sẽ được set là null ở những bộ bị xoá có did tương ứng )
* The Set-Null Policy: set null ở khoá ngoại did của relation Works .

2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.

CREATE TABLE Emp (

eid INTEGER PRIMARY KEY,

ename STRING,

age INTEGER,

salary REAL

);

CREATE TABLE Works (

eid INTEGER,

did INTEGER,

pct\_time INTEGER,

PRIMARY KEY (eid, did),

FOREIGN KEY (eid) REFERENCES Emp(eid),

FOREIGN KEY (did) REFERENCES Dept(did)

);

CREATE TABLE Dept (

did INTEGER PRIMARY KEY,

dname STRING,

budget REAL,

managerid INTEGER,

FOREIGN KEY (managerid) REFERENCES Emp(eid)

);

3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.

ALTER TABLE Dept

ADD CONSTRAINT chk\_manager\_not\_null

CHECK (managerid IS NOT NULL);

4. Write an SQL statement to add John Doe as an employee with *eid* = 101, *age* = 32 and *salary* = 15*,* 000

insert into Emp values(101,'John Doe',32,15000)

5. Write an SQL statement to give every employee a 10 percent raise

update Emp

set salary = salary \* 1.1;

6. Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

delete from Dept

where dname = 'Toy';

trong relation Works , tôi đã có một ràng buộc FOREIGN KEY (did) REFERENCES Dept(did) thì khi xoá tuple có dname là 'Toy' , công việc ở quan hệ Works có tham chiếu qua Dept có dname ='Toy' sẽ bị conflict. ( ràng buộc tham chiếu bị xung đột )

**Exercise 3.** Consider the following relations:



1. List all the foreign key constraints among these relations.

Các ràng buộc tham chiếu trong relation:

+ Enrolled:

* Enrolled ( sid) là khoá ngoại tham chiếu đến Students( sid)
* Enrolled ( cid) là khoá ngoại tham chiếu đến Courses( cid)

+Teaches:

* Teaches (fid) là khoá ngoại tham chiếu đến Faculty(fid)
* Teaches (cid) là khoá ngoại tham chiếu đến Courses(cid)

+ Meets\_In:

* Meets\_In (cid) là khoá ngoại tham chiếu đến Course(cid)
* Meets\_In (rno) là khoá ngoại tham chiếu đến Rooms(rno)

2. Give an example of a (plausible) constraint involving one or more of these relations that is not a primary key or foreign key constraint.

+ sid và cid trong relation Enrolled is not null

+cid và rno trong relation Meets\_In is not null

đảm bảo rằng cột **grade** trong bảng **Enrolled** chỉ chứa các giá trị hợp lệ như 'A', 'B', 'C', 'D',:

ALTER TABLE Enrolled

ADD CONSTRAINT check\_max\_courses

CHECK (grade IN ('A', 'B', 'C', 'D', 'F'));

3. Create the above database with their constraints.

create table Students (

sid char(100) primary key,

name varchar(100) not null,

login char(100) unique,

age int,

gpa real

);

go

create table Faculty(

fid char(100) primary key,

fname varchar(100) not null,

sal real,

);

go

create table Courses(

cid char(100) primary key,

cname varchar(100) not null,

creadits int ,

);

go

create table Rooms (

rno int primary key,

address varchar(300) not null,

capacity int,

);

go

CREATE TABLE Enrolled (

sid CHAR(100),

cid CHAR(100),

grade VARCHAR(2),

PRIMARY KEY (sid, cid),

FOREIGN KEY (sid) REFERENCES Students(sid),

FOREIGN KEY (cid) REFERENCES Courses(cid),

);

create table Teaches(

fid char(100),

cid char(100),

primary key ( fid, cid),

foreign key (fid ) references Faculty ( fid),

foreign key (cid ) references Courses ( cid),

);

go

create table Meets\_In(

cid char(100),

rno int,

time char(100),

primary key (cid, rno,time),

foreign key (cid) references Courses (cid),

foreign key ( rno ) references Rooms (rno)

);

Go

ALTER TABLE Enrolled

ADD CONSTRAINT check\_max\_courses

CHECK (grade IN ('A', 'B', 'C', 'D', 'F'));

**Exercise 4.** Consider the following relational schema and briefly answer the questions that follow:

Emp(*eid:* integer, *ename:* string, *age:* integer, *salary:* real)  
Works(*eid:* integer, *did:* integer, *pct\_time:* integer)  
Dept(*did:* integer, *budget:* real, *managerid:* integer)

1. Define a constraint on Emp that will ensure that every employee makes at least $10,000.

ALTER TABLE Emp

ADD CONSTRAINT check\_salary\_min

CHECK (salary >= 10000);

1. Define a constraint on Dept (using trigger) that will ensure that all managers have *age >* 30

CREATE TRIGGER CheckManagerAge

ON Dept

AFTER INSERT, UPDATE

AS

BEGIN

-- Check if any new manager's age is less than or equal to 30

IF EXISTS (

SELECT 1

FROM inserted i INNER JOIN Emp e ON i.managerid = e.eid

WHERE e.age <= 30

)

BEGIN

RAISERROR('Managers must be older than 30 years.', 16, 1);

ROLLBACK TRANSACTION; -- Rollback the transaction to prevent the change

END

END;

go

1. Write SQL statements to delete all information about employees whose salaries exceed that of the manager of one or more departments that they work in. Be sure to ensure that all the relevant integrity constraints are satisfied after your updates.

-- Tạo bảng tạm thời để lưu danh sách eid cần xóa

CREATE TABLE #TempTable (eid INT);

-- Chèn các giá trị eid cần xóa vào bảng tạm thời từ truy vấn

INSERT INTO #TempTable

SELECT e.eid

FROM Emp e

INNER JOIN Works w ON e.eid = w.eid

WHERE e.salary > (

SELECT salary

FROM Dept d

WHERE d.did = w.did

);

-- Xóa các bản ghi tương ứng trong bảng Works

DELETE FROM Works WHERE eid IN (SELECT eid FROM #TempTable);

-- Xóa các bản ghi tương ứng trong bảng Emp

DELETE FROM Emp WHERE eid IN (SELECT eid FROM #TempTable);

-- Xóa bảng tạm thời sau khi sử dụng xong

DROP TABLE #TempTable;

**Exercise 5.** Discuss the strengths and weaknesses of the trigger mechanism. Contrast triggers with other integrity constraints supported by SQL.

Thảo luận về điểm mạnh và điểm yếu của cơ chế kích hoạt. Trình kích hoạt tương phản với các ràng buộc toàn vẹn khác được SQL hỗ trợ.

Trình kích hoạt là một thủ tục được gọi tự động để đáp ứng với một thay đổi được chỉ định đối với cơ sở dữ liệu. Ưu điểm của cơ chế kích hoạt bao gồm khả năng thực hiện một hành động dựa trên kết quả của một điều kiện truy vấn. Tập hợp các hành động có thể được thực hiện là tập hợp lớn các hành động mà ràng buộc toàn vẹn có thể thực hiện (tức là báo cáo lỗi). Các hành động có thể bao gồm gọi cập nhật mới, xóa hoặc chèn truy vấn, thực hiện các câu lệnh định nghĩa dữ liệu để tạo bảng hoặc dạng xem mới hoặc thay đổi chính sách bảo mật. Trình kích hoạt cũng có thể được thực thi trước hoặc sau khi thực hiện thay đổi đối với cơ sở dữ liệu (nghĩa là sử dụng dữ liệu cũ hoặc mới)

**Exercise 6.** Consider the following relational schema. An employee can work in more than one department; the pct\_time ﬁeld of the Works relation shows the percentage of time that a given employee works in a given department.

Emp(eid: integer, ename: string, age: integer, salary: real)

Works(eid: integer, did: integer, pct\_time: integer)

Dept(did: integer, budget: real, managerid: integer)

Write integrity constraints (domain, key, foreign key, or CHECK constraints; or triggers to ensure each of the following requirements, considered independently.

1. Employees must make a minimum salary of $1000.
2. Every manager must be also be an employee.
3. The total percentage of all appointments for an employee must be under 100%.
4. A manager must always have a higher salary than any employee that he or she

manages.

1. Whenever an employee is given a raise, the manager’s salary must be increased to

be at least as much.

1. Whenever an employee is given a raise, the manager’s salary must be increased

to be at least as much. Further, whenever an employee is given a raise, the department’s budget must be increased to be greater than the sum of salaries of all employees in the department.

--1.

alter table Emp

add constraint Emp\_MinSalary

check (Emp.salary >= 1000)

--2.

alter table Dept

add constraint Valid\_manager

foreign key (managerid) references Emp(eid)

--3

alter table Works

add constraint pct\_check

check (pct\_time < 100)

IF OBJECT\_ID ('Sales.reminder1', 'TR') IS NOT NULL

DROP TRIGGER Sales.reminder1;

GO

CREATE TRIGGER reminder1 ON Sales.Customer

AFTER INSERT, UPDATE

AS RAISERROR ('Notify Customer Relations', 16, 10);

GO

--4

IF OBJECT\_ID ('Emp.CheckSalaryEmp', 'TR') IS NOT NULL

DROP TRIGGER Emp.CheckSalaryEmp;

GO

create trigger CheckSalaryEmp

on Emp

after insert, update

as

begin

if exists (

select 1

from Emp e,Works

where e.eid = Works.eid and e.salary >= (select Emp.salary from Emp,Dept where Dept.did = Works.did and Dept.managerid = Emp.eid )

)

begin

RAISERROR('A manager must always have a higher salary than any employee that he or she manages.',16,1);

ROLLBACK TRANSACTION;

end

end

--5

use COMPANY

go

DROP TRIGGER IncreaseSalary;

go

create trigger IncreaseSalary

on Emp

after update

as

if UPDATE(salary)

begin

-- lấy id và salary của thằng nhân viên vừa update #salary

declare @id int, @EmpSal real

select @id = new.eid, @EmpSal = new.salary

from inserted new

-- tìm ra id manager của nó

declare @managerid int

select @managerid = Dept.managerid

from Dept

where Dept.did = (select Works.did from Works where @id = Works.eid)

--Emp(eid) --> (eid)Works(did) --> (did)Dept

-- tăng lương manager nếu có

if @EmpSal > (select Emp.salary from Emp where Emp.eid = @managerid )

begin

-- Update Emp tại dòng có id của manager --> update salary

update Emp

set Emp.salary = @EmpSal

where Emp.eid = @managerid

end

end

--6

use COMPANY

go

DROP TRIGGER IncreaseSalary;

go

create trigger IncreaseSalary

on Emp

after update

as

if UPDATE(salary)

begin

-- lấy id và salary của thằng nhân viên vừa update #salary

declare @id int, @EmpSal real

select @id = new.eid, @EmpSal = new.salary

from inserted new

-- tìm ra id manager của nó

declare @managerid int

select @managerid = Dept.managerid

from Dept

where Dept.did = (select Works.did from Works where @id = Works.eid)

--Emp(eid) --> (eid)Works(did) --> (did)Dept

-- tăng lương manager nếu có

if @EmpSal > (select Emp.salary from Emp where Emp.eid = @managerid )

begin

-- Update Emp tại dòng có id của manager --> update salary

update Emp

set Emp.salary = @EmpSal

where Emp.eid = @managerid

end

-- tăng budget của dept nếu có

-- sum Lương của nhân viên trong bộ phân

-- did - dept

-- eid's did

-- @id--> (eid)Work(eid)-->(eid)Emp

-- tìm ra dept của Emp

declare @EmpDept int

select @EmpDept = Works.did from Works where Works.eid = @id

--EmpDept--> (did)Works(eid) --> (eid)Emp

declare @TotalSalary real

select @TotalSalary = sum(Emp.salary)

from Emp,Works

where Works.did = @EmpDept and Works.eid = Emp.eid

-- Dept budget update ( nếu có )

if @TotalSalary > ( select Dept.budget from Dept where Dept.did = @EmpDept)

begin

update Dept

set budget = @TotalSalary

where Dept.did = @EmpDept

end

end

go

**Exercise 7.** Brieﬂy answer the following questions based on this schema:

Emp(eid: integer, ename: string, age: integer, salary: real)

Works(eid: integer, did: integer, pct time: integer)

Dept(did: integer, budget: real, managerid: integer)

1. Suppose you have a view SeniorEmp deﬁned as follows:

CREATE VIEW SeniorEmp (sname, sage, salary)

AS SELECT E.ename, E.age, E.salary

FROM Emp E

WHERE E.age > 50

Explain what the system will do to process the following query:

SELECT S.sname

FROM SeniorEmp S

WHERE S.salary > 100,000

Truy vấn này trích xuất tên nhân viên Senior có tuổi lớn hơn 50 và có mức lương lớn hơn 100,000 từ view SeniorEmp

2. Give an example of a view on Emp that could be automatically updated by updating Emp.

--create view ( can update)

create view SeniorEmpSalary (sid,salary)

as select Emp.eid,Emp.salary

from Emp

where Emp.age >=35;

go

A screenshot of a computer

Description automatically generated

-- update view

update SeniorEmpSalary

set salary /=2;

A screenshot of a computer

Description automatically generated

3. Give an example of a view on Emp that would be impossible to update (automatically) and explain why your example presents the update problem that it does.

-- create view (un-update)

create view TotalSeniorSalary (salary)

as select sum(Emp.Salary) as ToTalSalary

from Emp

where Emp.age >=35;

go

A screenshot of a computer

Description automatically generated

update TotalSeniorSalary

set salary \*=2

A close up of a screen

Description automatically generated

Bởi vì nó là un-update view, view trên sử dụng hàm gộp sum để tính tổng lương , nên view đó là un-update view